Jammming Deezer Design Document

August 29th 2017 (JP Venter)

OBJECTIVE

To give users the ability to browse tracks by genre.

BACKGROUND

A user may just be interested in browsing tracks connected to particular genre's. Some genre examples include Pop, Dance, Rock, R&B etc.

This feature accomplishes the following:

- Display a list of all the genre's available in Deezer
- A user can click on a genre and have all the tracks in that genre returned in a separate list.
- A user can click on a track and have further info returned about that track such as Album cover, duration, artist, publish date etc.
- A user can add the track to or remove it from their favourites.

TECHNICAL DESIGN

1) **Retrieve and Display Deezer Genre's**

A new component ***GenresList*** should be created. This component, on render, will retrieve a list of Deezer genre's

The ***GenreList*** will use a ***GenreType*** component to render each individual genre.

We will need to initialize a state for ***GenreList*** to contain a key for ***GenreType*** that defaults to an empty array.

To retrieve genres, we will create a new method, ***Deezer.getGenres()***, that hits the http://api.deezer.com/genre endpoint using a GET request.

The empty ***GenreType*** array should then be populated with the results from the request. The array should have the following keys: ***id (int), type:genre name (string), picture:image url (string)***.

setState() should be called to have the nested ***GenreList*** component re-render.

The ***GenreList*** component should have a property called ***genreData*** assigned to the ***GenreType*** array.

2) Retrieve and Display Tracks from a Chosen Genre

The ***GenreType*** items should have an onClick event handler attached which references a function called ***handleGenreClick(event, genre_id).*** This function should accept an event object and the current target ***id*** value.

On click this method should hit the Deezer api - ***http://api.deezer.com/genre/{genre_id}/artists*** and return the data containing all artists linked to that genre.

A new component ***GenresArtistList*** should be created. This component, on render, will retrieve a list of Deezer artists associated with a genre id.

The ***GenresArtistList*** will use a ***GenreArtist*** component to render each individual genre.

We will need to initialize a state for ***GenresArtistList*** to contain a key for ***GenreArtist*** that defaults to an empty array.

3) Click on a track and view more information on that track

A details link would be present on each track. This link when clicked will call a method ***trackDetail(track_id)*** to display more information about that track. This method should hit the Deezer api with a fetch call to http://api.deezer.com/track/{track_id}.

The returned data details should be displayed in a new component called ***TrackDetail.*** TrackDetail should show artist, album and duration and be in the format of a dismissable modal popup.

4) Add an Artist to the Users Favourites

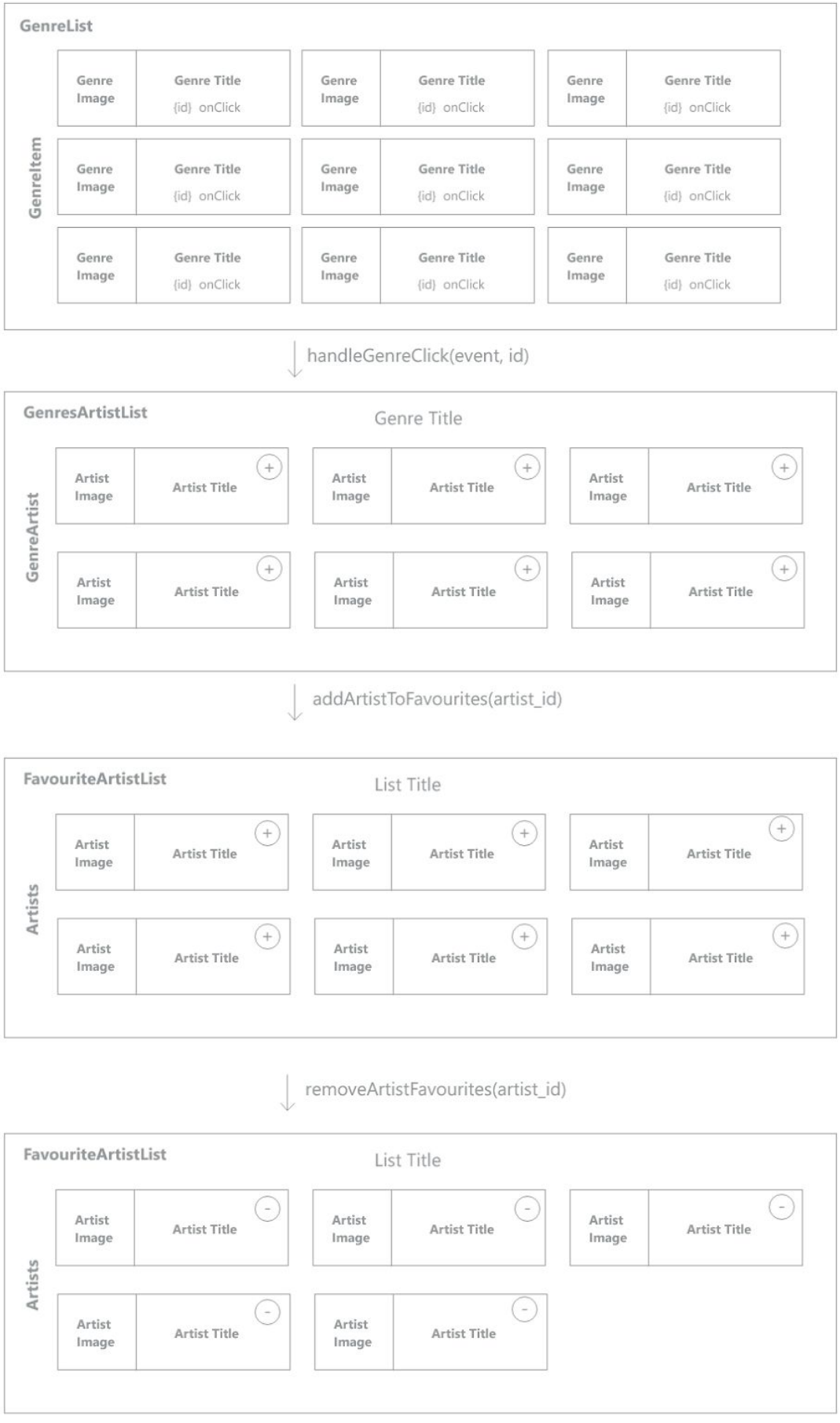
Each rendered ***GenreArtist*** will have the ability to add the artist to a new component, ***FavouriteArtistLists***. This list should be initiated with a state variable called **artists** pointing to an empty array.

An onClick method ***addArtistToFavourites(artist_id)*** will be attached to each rendered ***GenreArtist*** , this will call the Deezer end point with http://api.deezer.com/user/{user_id}/artists/{artist_id}. This will be a POST method.

The ***addArtistToFavourites(artist_id)*** will require the user_id prior to initiating the save call.

To retrieve the **user_id**, we will hit the http://api.deezer.com/user/me endpoint. We will create a new method called **getCurrentUserId()**. At the top of *Deezer.js*, we will instantiate a variable called **userId** with no value. Then inside ***getCurrentUserId()***, we will check to see if ***userId*****'s** value is already set (from a previous call to the function). If it is, we will create and return a promise that will resolve to that value. Otherwise we will make the call to the ***/me*** endpoint and return that request's promise.

Once our ***getCurrentUserId()*** is written, we should use it in ***addArtistToFavourites(artist_id)***. We will need to wait for the userId to be returned before proceeding with the save part of the method.


GenreList
GenreItem
Genre Image
Genre Title
{id} onClick
handleGenreClick(event, id)
GenresArtistList
Genre Title
GenreArtist
Artist Image
Artist Title
addArtistToFavourites(artist_id)
FavouriteArtistList
List Title
Artists
removeArtistFavourites(artist_id)

CAVEATS

Excess Favourites Saves

Each time an artist is added to the users favourites list, the ***addArtistToFavourites(artist_id)*** is called. This may send numerous requests to the Deezer api and cause the app to be throttled. It may be worthwhile considering updating the user's favourites list client side and then have a commit or update button to effect the save of the new user favourites state in bulk.